移动边缘计算中基于改进拍卖模型的计算卸载策略 *

盛津芳，滕潇雨，李伟民，王　斌
(中南大学 信息科学与工程学院，长沙 410000)

摘　要：随着移动互联网业务的快速发展，增强现实、虚拟现实、超清视频等手机应用逐渐普及、IoT 应用不断涌现，计算能力和续航能力的不足成为限制智能终端设备成功支撑这些应用的主要瓶颈。针对这一现状，采用计算卸载的方式解决该问题，在多用户多移动边缘服务器的场景下，综合考虑智能设备性能和服务器资源，提出了一种基于改进拍卖算法的计算卸载策略。该策略包含两个主要阶段：卸载决策阶段，通过综合考虑计算任务自身大小、计算需求和服务器计算能力、网络带宽等因素，提出了卸载决策的依据；任务调度阶段，通过综合考虑计算任务的时间需求和 MEC 服务器计算性能提出了基于改进拍卖算法的任务调度模型。通过实验证明，提出的计算卸载策略能够有效地降低服务时延，减少智能设备能耗，改善用户体验。

关键词：移动边缘计算；计算卸载；拍卖算法；失败补偿
中图分类号：TP391　　doi: 10.19734/j.issn.1001-3695.2018.10.0860

Computational unloading strategy based on improved auction model in mobile edge computing

Shengjinfang, Teng Xiaoyu, Li Weimin, Wang Bin
(*School of Information Science & Engineering*, *Central South University*, *Changsha* 410000, *China*)

Abstract: With the rapid development of mobile Internet services, mobile applications such as augmented reality, virtual reality, and ultra clear video have become popular and IoT applications are emerging. The limited computing power and the lack of endurance of smart terminal devices are becoming more and more inadequate for these applications. Aiming at this situation, this paper proposes a computational offload strategy based on improved auction algorithm under the premise of combining intelligent device performance and server resources in the scenario of multi-user and multi-MEC server, the strategy consists of two important phases: the unloading decision-making phase. By considering the calculation task size, computing requirements and server computing power, network bandwidth and other factors, this paper proposes the basis for the uninstallation decision; In the task scheduling phase, by considering the time requirements of the computing task and the computing performance of the MEC server, this paper proposes a task scheduling model based on improved auction algorithm. The experiment proves that the computational offloading strategy proposed in this paper can effectively reduce the service delay, reduce the energy consumption of smart devices, and improve user satisfaction.

Key words: moving edge calculation; calculation unloading; auction algorithm; failure compensation

0　引言

随着增强现实、虚拟现实、在线游戏、超清视频等移动互联网业务飞速发展，以及群智感知、智能监控、环境监测等 IoT 应用不断涌现，对于移动设备的服务水平的要求越来越高，这些应用需要丰富的计算资源，同时伴随着高额的能耗。而移动智能设备由于体积限制导致计算能力和电池续航能力有限，承担这些应用已经无法达到用户的满意水平。解决这个问题的一个主要途径是将计算密集的任务卸载到附近资源更加丰富的服务器上,这种方法称为计算卸载[1]或者游牧服务[2]。该方法共包含迁移环境感知（服务器发现）、任务分割、迁移决策、任务上传、任务远端执行、结果返回六个步骤。其中最重要的两个环节是任务分割和迁移决策。任务分割侧重于细粒度划分，基于应用程序代码，对应用程序根据方法、功能等标准进行分割，如文献[3]的 MAUI、文献[4]的 ThinkAir 和文献[5]的 Phone2Cloud。迁移决策则是计算卸载技术的核心问题，重点关注是否进行计算卸载，决策依赖于时间开销和能源消耗等。文献[6, 7]从能源的角度考虑了卸载决策，而文献[8]则是从服务器资源和带宽方面考虑。

计算卸载最早应用于移动云计算（mobile cloud computing，MCC）[9~11]，将移动终端的计算任务卸载到传统的云数据中心上执行。以集中式计算和存储为主要特征的传统云计算模式进行计算卸载的主要障碍是：通过广域网（wide area network，WAN）到达（远程）云服务器所经历的延迟可能已经抵消了计算卸载所节省的时间。在许多实时移动应用（如在线游戏、语音识别、Facetime）中，用户体验质量会受到严重影响。针对这一缺陷，学者提出了让云服务器离用户更近一些的思想，于是引入了 cloudlet[12, 13]的概念，通过 Wi-Fi 接入固定服务器，进行计算卸载[14~16]。但是 cloudlet 服务器位置是固定的，数量是有限的，缺乏可用的固定服务器可能会限制 cloudlet 的适用性。针对 MCC 和 cloudlet 在计算卸载应用上的缺陷，欧洲组织 TROPIC 提出在小型小区的基站上提供云计算的能力。随后欧洲电信标准协会（European Telecommunications Standards Institute ，ETSI）提出移动边

收稿日期：2018-10-31；**修回日期：**2019-01-04　**基金项目：**国家科技重大专项项目（2017ZX06002005）

作者简介：盛津芳（1971-），女，湖南人，副教授，博士，主要研究方向为分布式系统、数据处理；滕潇雨（1993-），女，山东人，硕士研究生，主要研究方向为云计算、边缘计算(xiaoyu7heyd@163.com)；李伟民（1988-），湖南人，博士，主要方向云计算、边缘计算；王斌（1973-），男，山西人，教授，博士，主要研究方向为软件工程.

缘计算（mobile edge computing ，MEC）的概念，将 MEC 与小型小区基站的部署相结合，可实现移动设备与提供计算服务的云服务器在物理意义上的真正接近，从而降低应用的访问延迟[17]。

移动边缘计算作为未来 5G 通信的关键技术，最主要的应用是计算卸载[18]，很多学者对这方面的研究作出了重要的贡献。文献[19]研究了多通道无线干扰环境中的多用户计算卸载问题，采用博弈论方法以分布式方式实现有效的信道分配。文献[20]采用正交频分复用（OFDMA）的方式接入多用户、多 MEC 服务器系统，通过双层优化方法，将原始的 NP 难问题解耦为寻求功率和子载波分配以及上层任务卸载问题的低层问题。文献[21] 提出了一种启发式卸载决策算法（HODA），该算法是半分布式的、联合优化卸载决策和通信资源，以最大化系统效用。文献[22] 提出了一个基于李雅普诺夫优化理论的任务调度算法，结合任务卸载、设备—基站关联以及基站睡眠的优化调度问题，最小化设备和基站的整体能量消耗。文献[8]提出了一种自适应的顺序卸载任务的方法来解决这些问题，其中移动用户基于当前干扰环境和可用计算资源顺序地作出卸载决策，在减少延迟和能量消耗方面取得了良好效果。上述研究都对计算卸载技术的发展起到了重要的推动作用，但是也都有各自的侧重点。例如文献[19~21]侧重于研究信道复用、信道竞争对卸载决策的影响，并没有考虑服务器资源使用情况对计算卸载性能的影响；文献[6, 8, 22]侧重于研究 MEC 服务器资源和调度，并没有考虑任务本身特性（任务数据量大小、计算资源需求等）对计算卸载的影响。针对现有的研究现状，本文提出了一种基于改进拍卖算法的计算卸载策略。通过联合考虑计算任务自身大小、计算需求和服务器计算能力和网络带宽等因素，提出了卸载决策的依据；通过联合考虑计算任务的时间需求和 MEC 服务器计算性能提出了基于改进拍卖的任务调度模型。实验证明，本文提出的卸载策略能够有效地降低服务时延，减少智能设备能耗，改善用户体验。

1　基于改进拍卖的决策模型

针对移动设备计算能力无法满足计算密集型应用的现状，本文采用将移动设备的任务卸载到 MEC 服务器上的计算卸载方式解决该问题。计算卸载技术主要解决两个问题：任务是否应该卸载执行？当任务需要卸载执行时，选择哪一个虚拟机（virtual machine，VM）处理该任务？围绕着两个问题，本文提出了基于改进拍卖模型的计算卸载策略，该策略分为两个阶段：

a)卸载决策阶段，结合任务在本地运行和计算卸载的时间消耗和能量消耗来共同决策该任务是否需要卸载执行。

b)当任务需要卸载执行时，通过改进拍卖算法对任务进行调度，在合适的 VM 上执行任务，以达到全局最优，即在一段时间内，MEC 服务器可以执行更多的任务。同时，提出了竞拍失败补偿模型，通过考虑任务截止时间的影响，以对买方价格补偿的方式，保证任务调度的公平性。

1.1　卸载决策模型

对于是否进行计算卸载，主要的判断依据有两个，即是否能够减少任务运行时间和是否能够节省手机能耗。本文针对任务本地运行和进行卸载运行的时间消耗及能源消耗提出了相应的模型。

1)本地运行耗时及能耗模型

当任务在本地运行时，处理该任务所需要的时间 T_L 为

$$T_L = \frac{C_n}{V_L} \tag{1}$$

其中：C_n 为计算任务 n 所需计算资源，这里的计算资源用执行该任务需要的 CPU 周期数来表示，具体的获取方式可以通过代码实现（如 C++中使用 get_cycle_count()函数）；V_L 为本地 CPU 的执行速率。

任务在本地运行时，处理该任务所消耗的能量 E_L 为

$$E_L = T_L \cdot P_L \tag{2}$$

其中：P_L 为手机运算功率。

2)计算卸载运行耗时及能耗模型

当任务在 MEC 服务器上运行时，处理该任务所消耗的时间 T_{unload} 一共由三部分组成，即任务上传到服务器上的传输时间 T_{up}、任务在服务器上运行的时间 T_{exe}、任务结果下传到手机的时间 T_{down}，因此计算卸载耗时 T_{unload} 表示为

$$T_{unload} = T_{up} + T_{exe} + T_{down} \tag{3}$$

许多研究[20~24]都忽略了结果回传时间 T_{down} 对卸载时间的影响，这是因为大多数应用的输出比输入小得多，所以回传时间对总耗时影响不大。文献[20]对此进行了详细的分析。因此，本文将卸载时间定义为

$$T_{unload} = T_{up} + T_{exe} \tag{4}$$

其中：任务上传需要的传输时间 T_{up} 可以表示为

$$T_{up} = \frac{D_n}{W\log_2\left(1+\frac{P_{up}\cdot Los}{N}\right)} \tag{5}$$

其中：D_n 是计算任务需要上传的数据量；N 表示信道内部的高斯噪声功率；W 为信道带宽；P_{up} 是手机上传功率。Los 是信道增益，基于蜂窝无线环境的无线干扰模型[25]将 Los 定义为基于距离的函数 $Los = d^{-\alpha}$，α 取值为 4。

任务在 MEC 服务器上的运行时间 T_{exe} 表示为

$$T_{exe} = \frac{C_n}{V_{cloud}} \tag{6}$$

其中：V_{cloud} 为服务器 CPU 的执行速率。

根据式(1)(2)可得任务卸载时间为

$$T_{unload} = \frac{C_n}{V_{cloud}} + \frac{D_n}{W\log_2\left(1+\frac{P_{up}\cdot Los}{N}\right)} \tag{7}$$

任务在 MEC 服务器上运行时，手机所消耗的能量仅仅是任务上传时的消耗，所以进行计算卸载时，能耗 E_{up} 表示为

$$E_{up} = T_{up}\cdot P_{up} = \frac{D_n}{W\log_2\left(1+\frac{P_{up}\cdot Los}{N}\right)}\cdot P_{up} \tag{8}$$

3)卸载决策依据

当采用计算卸载的任务运行时间低于本地运行时间，能量消耗低于本地能耗时，采用计算卸载更能提高用户体验质量，即满足式(9)(10)时可以进行计算卸载。

$$T_{unload} < T_L \tag{9}$$

$$E_{up} < E_L \tag{10}$$

根据式(1)(2)(7)~(10)可以推导出式(11)(12)。

$$V_{cloud} > \frac{V_L C_n}{C_n - \frac{D_n V_L}{W\log_2\left(1+\frac{P_{up}\cdot Los}{N}\right)}} \tag{11}$$

$$W > \frac{V_L P_{up} D_n}{P_L C_n \log_2\left(1+\frac{P_{up}\cdot Los}{N}\right)} \tag{12}$$

式(11)(12)的含义是，当服务器提供的计算能力 V_{cloud} 能够

大于不等式(11)右边的计算数值，用户所处的网络环境带宽能够满足式(12)，就可以进行计算卸载。

对于任务 n 来说，它基于的手机计算速率 V_L、任务需要的计算资源 C_n、任务的数据量大小 D_n、手机的上传功率 P_{up} 都是可以获得的。Los 是基于距离的函数，只要获知手机与基站的距离就能计算得到，即对于式(11)(12)来说，未知的参数只有 N，在本文中将 N 根据文献[25]定义为-100 dBm。至此，不等式(11)(12)右半部分的值是可以计算得到的。

计算卸载的第一步是迁移环境感知，在此步骤中可以获得 MEC 服务器提供的计算能力 V_{cloud} 和所处环境的网络带宽 W，然后与不等式右边的计算结果比较。如果满足式(11)(12)就可以进行计算卸载。同时，用户所处的地理环境中可能分布着多个基站，在多基站的情况下，就要分别计算任务进行卸载运行的时间消耗和能源消耗，在满足计算卸载的基站中，选择 $T_{unload}+E_{up}$ 取值最小的那个基站进行计算卸载。

1.2 基于改进拍卖的任务调度

当任务要以卸载的方式运行时，下一步需要考虑的两个问题是：任务要卸载到哪一台虚拟机上处理？卸载任务的数量较多而服务器计算资源有限时，任务应该如何调度？针对这两个问题，本文提出了基于改进拍卖算法的任务调度策略。

1）拍卖模型及问题形式化

拍卖模型如图 1 所示。在拍卖模型中，有买方和卖方两种角色，卖方提供商品，买方参与符合自己需求的产品的拍卖环节，价高者得。

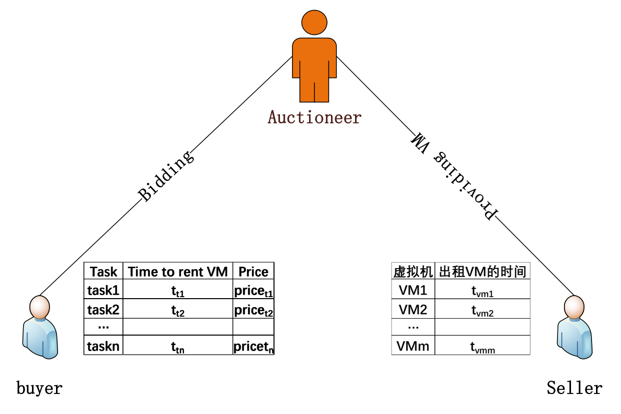


图 1 拍卖模型

Fig. 1 Auction model

图 1 中的卖方是指 MEC 服务器，销售的商品是计算资源的出租时间。这里的计算资源细化为 MEC 服务器中的空闲 VM，即卖方拍卖空闲 VM 的出租时间。买方指的是需要卸载运行的各个任务，买方的需求是在 VM 上运行该任务需要的时间即 T_{exe}，买方的出价标准是 $price=\frac{1}{T_L-T_{unload}}$。其中：$T_{unload}$ 为任务卸载耗时；T_L 为手机端运行该任务的耗时，两者相减为卸载执行任务的节约时间。同时也可以将 T_L 看做是完成该任务的截止时间，这样差值也代表完成任务的紧急程度。既然选择任务卸载，那就要尽可能地使任务卸载的时间 $T_L-T_{unload}>0$，即任务越接近截止时间，代表任务越紧急，所以取差值的倒数，剩余时间越少，它的倒数越大，意味着该任务愿意出更高的价格。

本文的拍卖模型中假定虚拟机的计算能力都是相同的，即用户租用虚拟机需要考虑的因素只有完成任务的时间。

定义一个时间片的时间为 s,用户租用 VM 的时间和服务器出租 VM 的时间都是 s 的整数倍。每轮时间片 s 进行一次拍卖，每次参与拍卖的用户既有上一轮拍卖失败的用户也有这一轮新加入的用户。

当拍卖开始时，参与拍卖的 VM 的出租时间为 t_{vm},任务需要租用虚拟机的时间为 t_t，只有满足 $t_t<t_{vm}$ 的任务才能参与竞争，价高者获得该 VM 的使用权，VM 执行该任务。等到出租时间截止时，取消该用户对该 VM 的使用权，虚拟机得以空闲，参与下一轮竞争。

如果仅仅按上文所述规则竞争虚拟机，在不理想的情况下，每一个虚拟机拍卖时，某些用户都满足 $t_t<t_{vm}$，但是由于这些用户出价低，有可能最多要参与 m 轮拍卖。这样在最坏的情况下，该算法的时间复杂度就是 O(nm)。针对原始拍卖算法时间复杂度较高的情况，本文提出了改进的拍卖算法。

2）改进拍卖算法

设买家集合为 TASK={task1,task2,…taskn},集合中的每一个 task 含有两个属性：taskn.t 表示完成任务 n 所需要的时间，taskn.p 表示任务 n 参与拍卖的出价。将集合中元素按照任务运行所需时间 task.t 升序排列。

设卖家的集合为 VM={vm1,vm2,…,vmm},集合中的每一个 vm 含有一个属性 vmn.t，表示虚拟机 n 的出租时间。将集合中元素按照出租时间 vm.t 升序排列。

定义集合 CA={task1,task2,…,taskm}用于存储满足拍卖条件，参与竞拍的任务。该集合的大小是有限的，最多存储 m 个（参与竞争的虚拟机数量）元素。元素插入集合的规则为：每插入一个元素，利用折半插入排序对集合中的元素按照任务出价降序排列。当参与竞争的任务数量大于 m 时，只取前 m 大的元素存入集合，剩余的任务意味着在竞争同一虚拟机时，它们的价格没有优势（只有 m 个虚拟机），因而竞拍失败。

遍历集合 VM，依次拍卖虚拟机。例如拍卖 vm1 时，依次遍历集合 TASK，将满足 taskn.t<vmn.t 的元素依次存入集合 CA 中，直到遍历到不能满足 taskn.t<vmn.t 的任务停止。然后取出集合第一个任务，将该 vm1 的使用权赋予该任务。再继续拍卖 vm2，vm3,…,vmm 最后 m 个虚拟机被 m 个用户竞拍到，其余竞拍失败用户等待下一轮竞拍。

3）拍卖补偿策略

本文中的竞拍价定义为 $\frac{1}{T_L-T_{unload}}$，虽然考虑了任务完成的紧急程度，但这也存在着不公平性。采用计算卸载更节省时间的任务在竞价上却失去了价格优势，对于这种任务在拍卖中是不公平的。因此本文提出了竞拍失败补偿策略。在任务竞价失败后，重新对该任务定价。

$$price=\frac{1}{T_L-T_{unload}-sX}$$

其中：s 是一个时间片的长度；X 为该任务参与竞拍失败的次数。该公式的含义之一是指该任务每参与一轮竞拍且竞拍失败，意味着任务截止时间就要减去一个时间片的长度（拍卖频率为每个时间片拍卖一次）；另一方面，对价格关于失败次数求导可得

$$price'=\frac{s}{(T_L-T_{unload}-sX)^2}$$

从导数公式中可以看出，随着 X 的增大，导数是递增的。导数越大，意味着价格增长速度越快。因此该公式的另一层含义是：随着竞拍失败次数的增加，对任务价格补偿就越多。

整个拍卖过程的算法描述如下：

算法 多任务多虚拟机一轮拍卖算法

Input：参与拍卖的任务和虚拟机。

Output：竞争失败的任务。

```
1: Calculate the time and bid for each task to rent a
virtual machine to get the set TASK.
2: According to the time required by the task, set the lease
time of the virtual machine to get the collection VM.
3: Ascending the elements in the collection TASK in
ascending order according to the lease time of the virtual
machine.
4: Ascending the elements in the collection VM in ascending
order according to the rental time of the virtual machine.
5: foreach vm in set VM
   foreach task in set TASK
     if taskn.t<vmn.t
       Add the task to the set CA
     else
       Add the task to the set EL
       // auction failed task set
     end foreach
     Assign the first task in the collection CA to the
virtual machine vm
   end foreach
   Add the remaining tasks in the set CA to the set EL
```

本文提出的多任务多虚拟机竞拍模型中，最核心的算法是拍卖竞争的环节，其中每一个 vm 参与拍卖，就要对集合 CA 进行一次折半插入排序，该折半插入排序算法的复杂度 $O(\log m)$,一共进行 m 轮拍卖，这意味着该算法的时间复杂度为 $O(m\log m)$，相比没有改进的拍卖算法（时间复杂度为 O(nm)），时间复杂度会有明显改善。因为随着移动智能终端的爆炸式增长，在未来参与计算卸载的任务数 n 肯定会远远大于提供计算资源的 VM 的数量 m。并且相对于无序的虚拟机参与拍卖，本文提出的将虚拟机集合和任务集合先升序排列再竞争的方式，会使得用户请求的时间和虚拟机出租的时间差别不大，这就意味着虚拟机在完成计算任务后，能够很快被回收，参与下一轮竞争。对于服务器来说，同样时间内它会处理更多的任务。

2　模拟实验及结果分析

在本节中使用 MATLAB 仿真本文提出的卸载策略。模拟的场景如下：定义有 1~2 个基站，每个基站上的 MEC 服务器运行着 30 台虚拟机，虚拟机的计算能力设置为 10 GHz[26]。用户与基站之间的通信带宽是 5 MHz[19]，并且用户在同一时刻只有一个计算任务。

本文采用人脸识别[26]算法作为计算任务，任务数据量大小为 1 000 KB~5 000 KB 随机分布，需要的计算资源设置为 200 Megacycles 到 1 000 Megacycles。通过调查，将用户手机的计算能力设置为 1.5 GHz 到 2.5 Ghz 随机分布。虚拟机的出租时间根据任务的需求时间，在最大值 GN 最小值之间做均匀分布。文中所有实验数据均是 20 次实验的平均值。

2.1　任务卸载计算和本地执行的手机能耗比较

本小节中，设置的模拟实验场景是 1 个基站/30 台虚拟机，参与卸载的决策的任务数是 20~70。图 2、3 分别展示了在用户距离基站 50、70、100 m 时参与卸载计算的任务数和手机端能耗。

从图 2 可以看出，当手机距离基站越近时，能够进行计算卸载的任务数越多，因为距离越近，无线通信的带宽损失越少。这也说明了边缘计算相对于传统的采取集中式运算的云计算来说，更适合计算卸载。图 3 则展示了任务全部在手机端运行和进行计算卸载时手机端的能耗，其中计算卸载的能耗由两部分组成，一部分是不能进行计算卸载的任务运行能耗和进行计算卸载的任务的上传能耗。从图中可以看出，对于手机来说，采用计算卸载方式处理任务能够至少节省 50%的电量，这就意味着手机有更长时间的待机能力。

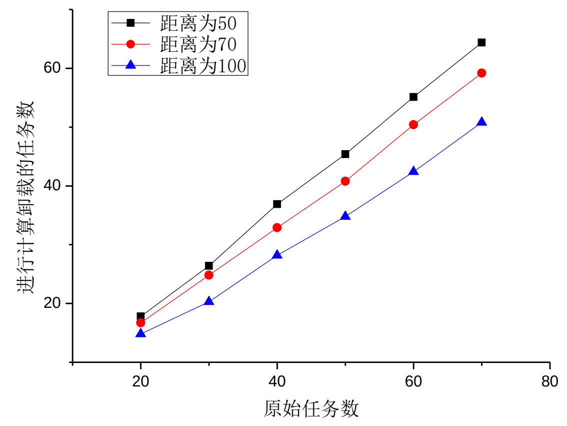


图 2　不同距离计算卸载的任务数

Fig. 2　Calculate the number of unloaded tasks at different distances

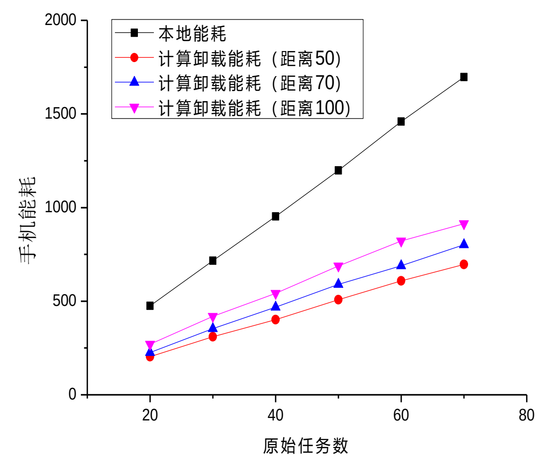


图 3　不同情况下手机端能耗

Fig. 3　Mobile phone energy consumption under different conditions

2.2　改进拍卖算法和经典拍卖算法的性能比较

本小节设置的实验场景是：有 2 个基站，共 60 个虚拟机。每一轮时间片产生的任务数是 20~70，一次实验进行 10 轮时间片。

图 4 中展示了基于改进拍卖算法和经典拍卖算法进行任务调度时的虚拟机空闲时间。本文中将虚拟机的空闲时间定义为虚拟机完成计算任务的时刻与下一轮时间片到来的时刻的差值。从图中可以看出，基于改进的拍卖算法能够有效地减少虚拟机的空闲时间。这种情况产生的原因是，在经典的拍卖算法中当一个虚拟机被拍卖时，所有运行时间满足条件的任务都会参与竞争，该虚拟机有可能被出价高，但需要运行时间很少的任务获得。简言之，就是出租时间很长的虚拟机与需要运行时间很短的任务绑定。这种情况下，产生较长的虚拟机空闲时间。而在本文提出的改进拍卖算法中，在拍卖前，将虚拟机与任务分别按出租时间，运行时间升序排列。拍卖时按照虚拟机出租时间从短到长依次拍卖,也就是说出租时间短的虚拟机被需求时间短的任务竞争，出租时间长的虚拟机被需求时间长的任务竞争，这样就会有效地缩短虚拟机的空闲时间。因此本文提出的改进拍卖算法不仅在时间复杂度上有优势，同时也提高了虚拟机效率。

图 5 展示了基于改进拍卖算法和经典拍卖算法进行任务调度时，当任务量远远大于虚拟机个数时，进行计算卸载的

任务中失败的任务数。本文中将失败的任务定义为：通过计算卸载耗时大于本地执行耗时的任务。从图中可以看出，基于改进拍卖的调度算法的失败任务数比未改进的低，这是因为基于改进拍卖的调度算法中虚拟机的空闲时间低，也就是说该算法中的虚拟机利用率会更高。

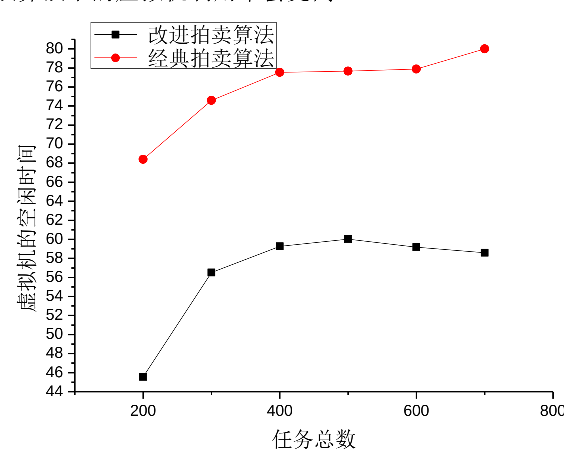


图 4 不同算法虚拟机空闲时间

Fig. 4 Different algorithm virtual machine idle time

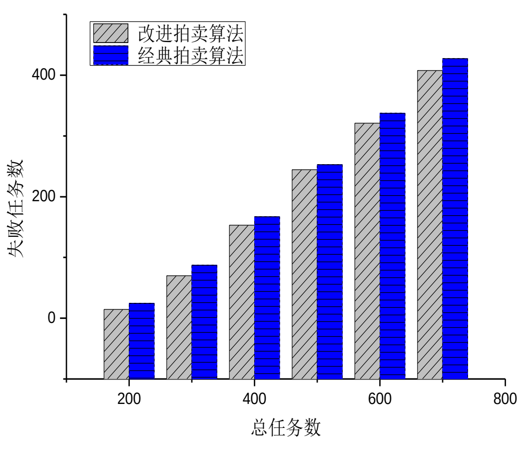


图 5 不同算法任务失败数

Fig. 5 Number of failures of different algorithm tasks

2.3 基于改进拍卖的卸载决策与其他方式的时间消耗比较

本小节设置的时间场景是：有 1 个基站，共 30 个虚拟机，每一轮时间片产生的任务数是 20~70。

图 6 展示了基于不同算法时，整个系统的时间开销比较。本文中将整个系统的时间开销定义为三部分：本地执行的任务的消耗的时间、进行计算卸载的任务的时间以及任务响应的时间（竞拍失败时，响应时间增加）。从图中可以看出，基于计算卸载比基于本地执行在时间上有着明显的优势。随着用户数量的增多，进行计算卸载比全部在本地运行的时间开销减少了将近 50%。同时通过改进拍卖算法，使得系统的计算开销进一步降低。实验数据证明，本文提出的基于改进拍卖算法的计算卸载策略能够有效地减少服务时延，提高用户满意度。

3 结束语

针对移动设备计算能力无法满足计算密集型应用的现状，本文采用将移动设备的任务卸载到移动边缘服务器上的计算卸载方式解决改问题，提出了基于改进拍卖算法的计算卸载策略，通过考虑计算任务自身需求和服务器计算能力、网络带宽等因素，提出了卸载决策的依据。通过联合考虑计算任务的时间需求和 MEC 服务器计算性能提出了基于改进拍卖的任务调度模型。并且通过实验验证，证明本文提出的卸载策略能够有效地提高任务执行速度，减少服务时延，提高手机续航能力，提升用户满意度。

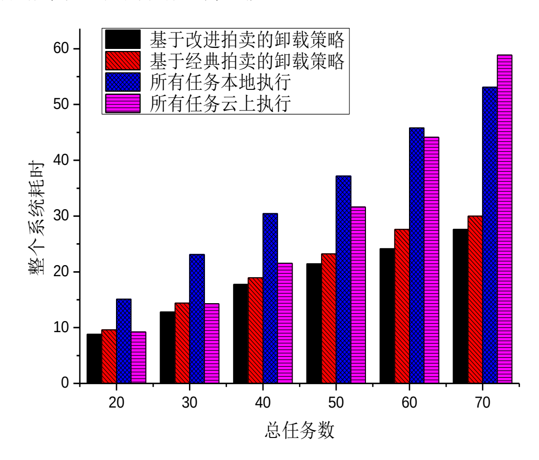


图 6 不同算法整个系统耗时

Fig. 6 Different algorithms time consuming for the entire system

参考文献：

[1] Kumar K, Liu Jibang, Lu Y H, *et al.* A survey of computation offloading for mobile systems [J]. Mobile Networks & Applications, 2013, 18 (1): 129-140.

[2] Sharifi M, Kafaie S, Kashefi O. A survey and taxonomy of cyber foraging of mobile devices [J]. IEEE Communications Surveys & Tutorials, 2012, 14 (4): 1232-1243.

[3] Cuervo E, Balasubramanian A, Cho D, *et al.* MAUI: making smartphones last longer with code offload [C]// Proc of International Conference on Mobile Systems, Applications, and Services. 2010: 49-62.

[4] Kosta S, Aucinas A, Hui Pan, *et al.* ThinkAir: dynamic resource allocation and parallel execution in the cloud for mobile code offloading [C]// Proc of IEEE INFOCOM. 2012: 945-953.

[5] Xia Feng, Ding Fangwei, Li Jie, *et al.* Phone2Cloud: exploiting computation offloading for energy saving on smartphones in mobile cloud computing [J]. Information Systems Frontiers, 2014, 16 (1): 95-111.

[6] Mao Yuyi, Zhang Jun, Letaief K B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices [J]. IEEE Journal on Selected Areas in Communications, 2016, 34 (12): 3590-3605.

[7] You Changsheng, Huang Kaibin, Chae H, *et al.* Energy-efficient resource allocation for mobile-edge computation offloading [J]. IEEE Trans on Wireless Communications, 2017, 16 (3): 1397-1411.

[8] Deng Maofei, Tian Hui, Lyu Xinchen. Adaptive sequential offloading game for multi-cell Mobile Edge Computing [C]// Proc of IEEE International Conference on Telecommunications. 2016: 1-5.

[9] Fernando N, Fernando N, Loke S W, *et al.* Mobile cloud computing: a survey [J]. Future Generation Computer Systems, 2013, 29 (1): 84-106.

[10] 李继蕊，李小勇，高云全，等. 5G 网络下移动云计算节能措施研究 [J]. 计算机学报, 2017, 40 (7): 1491-1516. (Li Jirui, Li Xiaoyong, Gao Yunquan, *et al.* Research on energy saving measures of mobile cloud computing under 5G network [J]. Chinese Journal of Computers, 2017, 40 (7): 1491-1516.)

[11] Sanaei Z, Abolfazli S, Gani A, *et al.* Heterogeneity in mobile cloud computing: taxonomy and open challenges [J]. IEEE Communications

Surveys & Tutorials, 2013, 16 (1): 369-392.

[12] Satyanarayanan M, Bahl P, Caceres R, *et al.* The case for VM-based cloudlets in mobile computing [J]. IEEE Pervasive Computing, 2009, 8 (4): 14-23.

[13] 赵梓铭，刘芳，蔡志平，等. 边缘计算：平台、应用与挑战 [J]. 计算机研究与发展, 2018, 55 (2): 327-337. (Zhao Yuming, Liu Fang, Cai Zhiping, *et al.* Edge computing: platform, application and challenges [J]. Journal of Computer Research and Development, 2018, 55 (2): 327-337.)

[14] 施巍松，孙辉，曹杰，等. 边缘计算：万物互联时代新型计算模型 [J]. 计算机研究与发展, 2017, 54 (5): 907-924. (Shi Yusong, Sun Hui, Cao Jie, *et al.* Edge computing: a new computing model in the age of internet of things [J]. Journal of Computer Research and Development, 2017, 54 (5): 907-924.)

[15] Zhang Yang, Niyato D, Wang Ping. Offloading in mobile cloudlet systems with intermittent connectivity [J]. IEEE Trans on Mobile Computing, 2015, 14 (12): 2516-2529.

[16] Jia Mike, Cao Jiannong, Liang Weifa. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks [J]. IEEE Trans on Cloud Computing, 2017, PP (99): 1-1.

[17] Sardellitti S, Scutari G, Barbarossa S. Joint optimization of radio and computational resources for multicell mobile-edge computing [J]. IEEE Trans on Signal and Information Processing over Networks, 2015, 1 (2): 89-103.

[18] 田辉，范绍帅，吕昕晨，等. 面向 5G 需求的移动边缘计算 [J]. 北京邮电大学学报, 2017, 40 (2): 1-10. (Tian Hui, Fan Shaoshuai, Lyu Yichen, *et al.* Mobile edge computing for 5G demand [J]. Journal of Beijing University of Posts and Telecommunications, 2017, 40 (2): 1-10.)

[19] Chen Xu, Jiao Lei, Li Wenzhong, *et al.* Efficient multi-user computation offloading for mobile-edge cloud computing [J]. IEEE/ACM Trans on Networking, 2016, 24 (5): 2795-2808.

[20] Cheng Kang, Teng Yinglei, Sun Weiqi, *et al.* Energy-efficient joint offloading and wireless resource allocation strategy in multi-mec server systems [EB/OL]// https://arxiv. org/abs/1803. 07243. (2018-03-20) [2018-10-31].

[21] Lyu Xinchen, Tian Hui, Zhang Pin, *et al.* Multiuser joint task offloading and resource optimization in proximate clouds [J]. IEEE Trans on Vehicular Technology, 2017, 66 (4): 3435-3447.

[22] 于博文，蒲凌君，谢玉婷，等. 移动边缘计算任务卸载和基站关联协同决策问题研究 [J]. 计算机研究与发展, 2018, 55 (3): 537-550. (Yu Bowen, Pu Lingjun, Xie Yuting, *et al.* Research on mobile edge computing task offloading and base station association collaborative decision making problem [J]. Journal of Computer Research and Development, 2018, 55 (3): 537-550.)

[23] Lin Xiaopeng, Zhang Heli, Ji Hong, *et al.* Joint computation and communication resource allocation in mobile-edge cloud computing networks [C]// Proc of IEEE International Conference on Network Infrastructure and Digital Content. 2017: 190-195.

[24] Huang Dong, Wang Ping, Niyato D. A dynamic offloading algorithm for mobile computing [J]. IEEE Trans on Wireless Communications, 2012, 11 (6): 1991-1995.

[25] Rappaport T. Wireless communications: principles and practice [M]. 电子工业出版社, 2013. (Rappaport T. Wireless communications: principles and practice [M]. Publishing House of Electronics Industry, 2013.)

[26] Soyata T, Muraleedharan R, Funai C, *et al.* Cloud-vision: real-time face recognition using a mobile-cloudlet-cloud acceleration architecture [C]// Proc of IEEE Computers and Communications. 2012: 59-66.